Make the Most of Your Inheritance with the SAS® ODS Styles.Default Lineage Tracer

Perry Watts, Independent Consultant, Elkins Park, PA

ABSTRACT

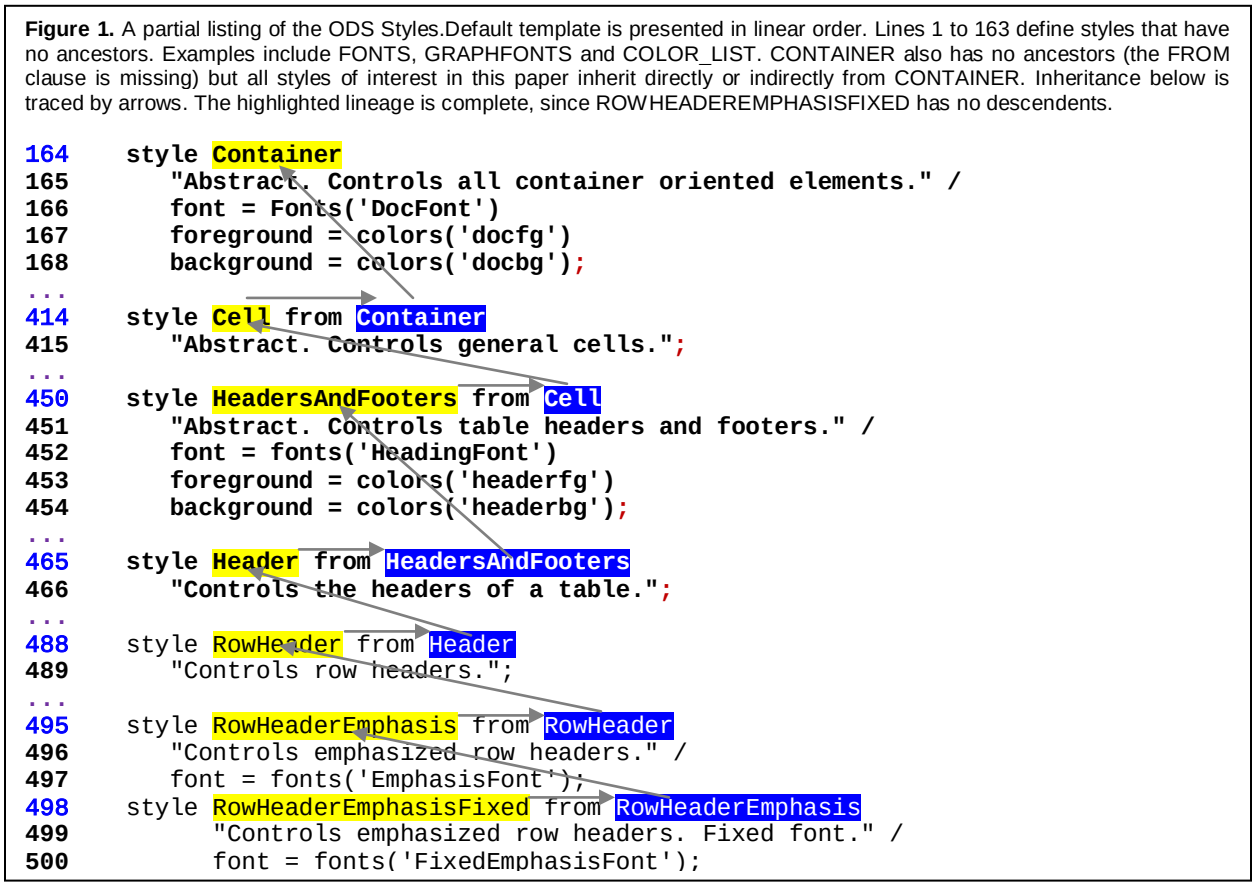
The ODS Styles.Default lineage tracer lists the sixty lineages of Container style elements that are defined in the ODS Styles.Default template for version 9.1.3 SAS. A detailed listing that includes default assignments for associated attributes can be retrieved by drilling down on a selected lineage in the home page on the HTML file. By using this tool, along with several others, you can make inheritance work for you when you need to customize an ODS-generated table.

This paper shows by example how to use the lineage tracer to develop new styles from the default template. Inheritance and references to "abstract" style elements are described along with attribute definitions and their default assignments. The goal of the paper is to develop a more comprehensive context for style element definitions so that the guess-work can be taken out the customization process. Construction of the lineage tracer is described in a separate paper: *Using Recursion to Trace Lineages in the SAS® ODS Styles.Default Template* [4].

PROBLEM DEFINITION

There is no way to organize the Styles.Default template by lineage. Instead, members of any given lineage are widely scattered throughout the template. For example in Figure 1, lines 164 to 498 from the template must be scanned to locate the six ancestors for the ROWHEADEREMPHASISFIXED style element. The only structural rule that can be counted upon is that ancestors precede their descendents in the Styles.Default template.

Figure 1. A partial listing of the ODS Styles.Default template is presented in linear order. Lines 1 to 163 define styles that have no ancestors. Examples include FONTS, GRAPHFONTS and COLOR_LIST. CONTAINER also has no ancestors (the FROM clause is missing) but all styles of interest in this paper inherit directly or indirectly from CONTAINER. Inheritance below is traced by arrows. The highlighted lineage is complete, since ROWHEADEREMPHASISFIXED has no descendents.



```
164    style Container
165       "Abstract. Controls all container oriented elements." /
166       font = Fonts('DocFont')
167       foreground = colors('docfg')
168       background = colors('docbg');
...
414    style Cell from Container
415       "Abstract. Controls general cells.";
...
450    style HeadersAndFooters from Cell
451       "Abstract. Controls table headers and footers." /
452       font = fonts('HeadingFont')
453       foreground = colors('headerfg')
454       background = colors('headerbg');
...
465    style Header from HeadersAndFooters
466       "Controls the headers of a table.";
...
488    style RowHeader from Header
489       "Controls row headers.";
...
495    style RowHeaderEmphasis from RowHeader
496       "Controls emphasized row headers." /
497       font = fonts('EmphasisFont');
498    style RowHeaderEmphasisFixed from RowHeaderEmphasis
499          "Controls emphasized row headers. Fixed font." /
500          font = fonts('FixedEmphasisFont');
```

Because manual traces are so labor intensive, no SAS user paper has been written that deals substantively with inheritance from the ODS styles.default template. Without a lineage tracer it is just about impossible to ferret out any rules for inheritance. In Figure 2 below, the manual trace from Figure 1 is translated into Lineage #20.

Figure 2. The manual trace from Figure 1 becomes Lineage 20 in the Container Lineage Tracer. Drilling down on the green number in the home page of the HTML file brings up detailed information about lineage-associated attributes.

Lineage#	Style#1	Style#2	Style#3	Style#4	Style#5	Style#6	Style#7
20	Container	Cell	HeadersAndFooters	Header	RowHeader	RowHeaderEmphasis	RowHeaderEmphasisFixed

Detailed Listing for Lineage #20

Style	Default Assignment	Attribute
Container	**Abstract. Controls all container oriented elements.**	
	font = Fonts('DocFont')	FONT
	foreground = colors('docfg')	FOREGROUND
	background = colors('docbg');	BACKGROUND
Cell	**Abstract. Controls general cells.**	
HeadersAndFooters	**Abstract. Controls table headers and footers.**	
	font = fonts('HeadingFont')	FONT
	foreground = colors('headerfg')	FOREGROUND
	background = colors('headerbg');	BACKGROUND
Header	**Controls the headers of a table.**	
RowHeader	**Controls row headers.**	
RowHeaderEmphasis	**Controls emphasized row headers.**	
	font = fonts('EmphasisFont');	FONT
RowHeaderEmphasisFixed	**Controls emphasized row headers. Fixed font.**	
	font = fonts('FixedEmphasisFont');	FONT

Additional tools derived from the Lineage Tracer, the styles.default template, and the 9.1.3 ODS User's Guide help to illustrate how inheritance works in PROC TEMPLATE. Sample listings can be found in the Appendix:

1) A listing of 66 **attributes** are adapted from the User's Guide. 28 out of the 66 are colored in light gray to indicate they are not included in the styles.default template.

2) A color tracer that traces color settings: ***location*** → ***color abbreviation*** → ***color*** → a list of ***style elements*** that use the given color. Style elements polled for information include **color_list** and **colors**. (→ = 'map to'). Color settings assigned by inheritance to Container style elements are added to the tracer.

3) A font tracer that traces font settings: ***font*** → ***font name*** → ***style elements*** that use the font. The style element polled for information is aptly named ***fonts.*** Font settings assigned by inheritance to Container style elements are added to the tracer.

Colors and fonts require their own tracers, since over 50% of the 184 attributes associated with Container style elements in the styles.default template reference either colors or fonts.

INHERITANCE: ABSTRACT VS REGULAR STYLE ELEMENTS

Abstract and regular style elements are clearly distinguishable in the screen snapshot of the lineage tracer displayed in Figure 3. From Figure 3, it can be seen that all lineages contain at least one abstract style element and one regular element. All abstract style elements in a lineage precede their regular counterparts.

From *Output Delivery System: The Basics and Beyond*, abstract style elements are described as follows:

> Some of the ODS style elements inherit attributes from abstract style elements. *The only purpose of an abstract style element is to provide an element from which many other style elements can inherit their attributes* **[2, p. 370]** (italics added).

If the statement is to be taken at face value, abstract style elements should never be able to play the *child* role in a style element definition. This assumption is tested when STYLE and REPLACE statements are compared in the section that follows Figure 3.

Figure 3. Part of the cover page for the ODS Lineage Tracer. The tracer is contained in an HTML file separate from the paper.

60 Container Lineages in the ODS Styles.Default Template
(Abstract Classes are in Blue)

Lineage#	Style#1	Style#2	Style#3	Style#4	Style#5	Style#6	Style#7
1	Container	BylineContainer					
2	Container	Cell	Data	DataEmphasis	DataEmphasisFixed		
3	Container	Cell	Data	DataEmpty			
4	Container	Cell	Data	DataFixed			
5	Container	Cell	Data	DataStrong	DataStrongFixed		
6	Container	Cell	HeadersAndFooters	Caption	AfterCaption		
7	Container	Cell	HeadersAndFooters	Caption	BeforeCaption		
8	Container	Cell	HeadersAndFooters	Footer	FooterEmphasis	FooterEmphasisFixed	
9	Container	Cell	HeadersAndFooters	Footer	FooterEmpty		
10	Container	Cell	HeadersAndFooters	Footer	FooterFixed		
11	Container	Cell	HeadersAndFooters	Footer	FooterStrong	FooterStrongFixed	
12	Container	Cell	HeadersAndFooters	Footer	RowFooter	RowFooterEmphasis	RowFooterEmphasisFixed
13	Container	Cell	HeadersAndFooters	Footer	RowFooter	RowFooterEmpty	
14	Container	Cell	HeadersAndFooters	Footer	RowFooter	RowFooterFixed	
15	Container	Cell	HeadersAndFooters	Footer	RowFooter	RowFooterStrong	RowFooterStrongFixed
16	Container	Cell	HeadersAndFooters	Header	HeaderEmphasis	HeaderEmphasisFixed	
17	Container	Cell	HeadersAndFooters	Header	HeaderEmpty		
18	Container	Cell	HeadersAndFooters	Header	HeaderFixed		
19	Container	Cell	HeadersAndFooters	Header	HeaderStrong	HeaderStrongFixed	
20	Container	Cell	HeadersAndFooters	Header	RowHeader	RowHeaderEmphasis	RowHeaderEmphasisFixed

STYLE VS REPLACE WITH ABSTRACT STYLE ELEMENTS

An amended version of the sashelp.shoes data set supports all the examples presented in the paper. Here is how the data set has been changed:

```
data shoes2(keep=product subsidiary stores rename=(stores=nstores));
  set sashelp.shoes;
  if product in ('Boot','Sandal','Slipper','Sport Shoe');
run;

proc sort data=shoes2 out=styleApp.shoes2;
  by product subsidiary;
run;
```

Assumptions are tested in the tables below. Except for the first table, code for PROC TEMPLATE only is displayed along-side output. First up is the default output from the styles.default template. To validate color and font assignments, check Figure 2 plus the font and color tracers.

1) Default Listing

Code

```
ods path work.temp(update)
     sasuser.templat(update)
     sashelp.tmplmst(read);
ods html path="&htmPath" (url=none)
  file='default.html' style=STYLES.DEFAULT;
  title 'Default Template Output';
  proc freq data=styleApp.shoes2;
    weight nstores;
    tables product;
  run;
ods _all_ close;
```

HTML Output

Default Template Output

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Changing Abstract Style Elements with a STYLE Statement:

Headers are not changed with a STYLE statement, because HEADERSANDFOOTERS, an abstract style element, plays both the *child* and *parent* role in the STYLE definition (**`style child from parent`**).

2) Changing HeadersAndFooters Fails with a STYLE Statement

Code

```
proc template;
  define style styles.abstract_style;
  parent=styles.default;
 /* from lineage #20 */
  style HeadersAndFooters
   from headersAndFooters /
     font = (arial,12pt,bold)
     background=black
     foreground=white;
  end;
run;
```

HTML Output

Abstract Element Change Fails With a STYLE Statement

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Changes are not restricted to within-lineage attributes. However, the STYLE definition still fails when BORDERWIDTH is set to 5 pixels.

3) Changing an Out-Of-Lineage Attribute Still Fails with a STYLE Statement

Code

```
proc template;
  define style styles.abstract_style2;
  parent=styles.default;
 /* from lineage #20 */
  style HeadersAndFooters
   from headersAndFooters /
    borderwidth=5;
  end;
run;
```

HTML Output

Abstract Element Change Fails to Increase Border Widths

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Changing Abstract Style Elements with a REPLACE Statement:

ODS output can be customized when REPLACE is substituted for STYLE. Now row and column headers reflect the changes made to the attributes associated with HEADERSANDFOOTERS.

4) Changing an Abstract Style Element Succeeds with a REPLACE Statement

Code

```
proc template;
   define style styles.abstract_Replace;
   parent=styles.default;
  /* from lineage #20 */
   replace HeadersAndFooters /
     font = (arial,12pt,bold)
     background=black
     foreground=white;
   end;
 run;
```

HTML Output

Headers are Changed with REPLACE

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

REPLACE comes with some pitfalls, however. To avoid surprises, specify values for all attributes associated with an abstract style in the styles.default template. In Table 5 below, only the out-of-lineage attribute **`borderwidth`** is spe-

cified. As a result, default values from **Container** are assigned to **foreground, background,** and **font** in HEADERSANDFOOTERS.

5) Results are Surprising when Attribute Changes with REPLACE are Incomplete

Code

```
/* font, background, and foreground are not
specified. */
proc template;
  define style styles.abstract_replace2;
  parent=styles.default;
 /* from lineage #20 */
  replace headersAndFooters /
    borderwidth=5;
  end;
run;
```

HTML Output

Headers Inherit from CONTAINER
When Only BorderWidth is REPLACED

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

By consulting the lineage, font, and color tracers, HEADERSANDFOOTERS and CONTAINER can be compared:

```
DEFAULT HeadersAndFooters (See Table 3):
  font = fonts('HeadingFont') = "arial, helvetica, sans-serif",4,bold
  foreground = colors('headerfg') = cx0033aa (lighter blue)
  background = colors('headerbg') = cxb0b0b0 (darker gray)
DEFAULT Container (See Table5):
  font = Fonts('DocFont') = "arial, helvetica, sans-serif",3   (implied regular font-weight)
  foreground = colors('docfg') = cx002288  (darker blue)
  background = colors('docbg') = cxe0e0e0  (lighter gray)
```

Colors and fonts can be verified by matching output in Tables 3 and 5 with the attribute settings listed above. In Table 5, however, the emboldened header is unexpected, since "bold" is not in the font setting. It would appear that PROC FREQ takes precedence over ODS when it comes to applying font-weights to headers. To restore header defaults along with a border width, all attributes must be defined with REPLACE:

6) Default Fonts and Colors are Specified Along with a Border Width with REPLACE

Code

```
proc template;
  define style styles.abstract_replace3;
  parent=styles.default;
 /* from lineage #20 */
  replace headersAndFooters /
   font = fonts('HeadingFont')
   foreground = colors('headerfg')
   background = colors('headerbg')
   borderwidth=5px;
  end;
run;
```

HTML Output

Defaults are Restored and a BorderWidth
is Specified with REPLACE

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

An expanded **REPLACE headersAndFooters FROM headersAndFooters** would not work in Table 6, because REPLACE destroys HEADERSANDFOOTERS before it can be referenced by FROM! In this situation, ODS just traverses up the lineage and deposits CONTAINER defaults into the output.

STYLE VS REPLACE WITH REGULAR STYLE ELEMENTS

The regular style elements that are being considered in this section are HEADER and ROWHEADER. They have no attributes associated with them in the Styles.Default template. In the ODS User's Guide it is stated that there are two ways to modify existing style elements:

- change ***only*** the style element that you specify by using the STYLE statement (bold italics added)
- change the style element that you specify and all the style elements that inherit from that element by using the REPLACE statement **[3, p. 335].**

As we just saw, STYLE doesn't work with abstract style elements, but it does work as advertised with regular style elements. Again we illustrate inheritance for regular style elements using the shoes2 data set. The next example un-

derscores the range of the STYLE statement in ODS. Only the HEADER is changed. ROWHEADER retains the defaults from the Styles.Default template.

7) ROWHEADER does not Inherit from HEADER when STYLE is Used

Code

```
proc template;
  define style styles.Regular_style;
  parent=styles.default;
/* from lineage #20 */
  STYLE Header /
    font = (arial,12pt,bold)
    background=black
    foreground=white;
  end;
run;
```

HTML Output

RowHeader Doesn't Inherit from Header with the STYLE Statement

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

To get the same format for HEADER and ROWHEADER, use a REPLACE statement:

8) ROWHEADER Inherits from HEADER when REPLACE is Used

Code

```
proc template;
  define style styles.Regular_style2;
  parent=styles.default;
/* from lineage #20 */
  REPLACE Header /
    font = (arial,12pt,bold)
    background=black
    foreground=white;
  end;
run;
```

HTML Output

RowHeader Inherits from Header with the REPLACE Statement

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

THE SPECIAL CASE OF THE JUSTIFICATION ATTRIBUTES (VJUST AND JUST)

When border widths were set to 5 pixels in Table 6, we learned that the choice of attributes available for modification is not restricted by lineage affiliation. However, font settings in Table 5 hint at conflicts between ODS and formatting defaults set by other procedures. In Table 5, the header font was emboldened by PROC FREQ even though the default setting in ODS called for a medium font weight. Similar issues arise for the JUST and VJUST attributes in Tables 9 and 10 below. In Table 9, PROC FREQ dictates the header format whereas PROC PRINT calls for different formatting in Table 10.

9) PROC FREQ Determines the Justification of HEADERS and ROWHEADERS

Code

```
proc template;
  define style styles.HeadersJust;
  parent=styles.default;
/* FROM LINEAGE #20*/
  style header/
    font = (arial,12pt,bold)
    background=black  foreground=white
    cellwidth=1in  cellheight=0.8in
    vjust=top   /* GET BOTTOM */
    just=left;  /* GET RIGHT */
  style rowheader/
    font = (arial,12pt,bold)
    background=black  foreground=white
    cellheight=0.5in
    vjust=top   /* WORKS TOP, MIDDLE, BOTTOM */
    just=right; /* GET LEFT */
  end;
run;
```

HTML Output

JUST Fails for HEADER and ROWHEADER. CELLWIDTH and CELLHEIGHT Work. VJUST Yields Mixed Results

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

For verification, match code line comments to table output. Inconsistencies are noted between attribute assignments and the line comments. For example, even though JUST is set to LEFT for in the HEADER style statement, right justification is what occurs (except for “Product” which is left-justified like the other row headers).

In Table 10, an attempt is made to left-justify the row header inside ODS. However, since ROWHEADER is reserved for the observation number in PROC PRINT, right justification prevails. (This time “Obs” appears as a center-justified header, not a right-justified row header).

10) PROC PRINT Determines the Justification of HEADERS and ROWHEADERS

Code

```
proc template;
  define style styles.HeadersJust2;
  parent=styles.default;
/* FROM LINEAGE #20*/
  style header/
    font = (arial,12pt,bold)
    background=black  foreground=white
    cellwidth=1in  cellheight=0.8in
    vjust=top    /* WORKS TOP MIDDLE BOTTOM */
    just=right;  /* GET CENTER */
  style rowheader/
    font = (arial,12pt,bold)
    background=black  foreground=white
    cellheight=0.5in
    vjust=bottom /* WORKS TOP MIDDLE BOTTOM */
    just=left;   /* GET RIGHT */
  end;
run;
```

HTML Output

JUST Fails for HEADER and ROWHEADER
CELLWIDTH, CELLHEIGHT & VJUST Work

The Print Procedure

Obs	Product	Subsidiary	Number of Stores
1	Boot	Addis Ababa	12
2	Boot	Al-Khobar	10
3	Boot	Algiers	21
4	Boot	Auckland	12
5	Boot	Bangkok	1
6	Boot	Bogota	19
7	Boot	Budapest	22

THE ROLE OF *FROM* IN A STYLE ELEMENT DEFINITION

A *from* clause has to be used when the list of lineage attributes being modified in a given STYLE or REPLACE statement is incomplete. In this situation, default values from the *from* style element supply the missing settings. From figure 2, it can be seen that three attributes are defined in Lineage #20. They are **font**, **foreground** and **background**. In the examples that follow no more than two out of the three attributes are assigned in any single style statement.

The most common form of inheritance is for the *child* element to inherit default settings for attributes from its immediate ancestor, the *parent*. In Table 11, only the **font** and **foreground** attributes are being changed for HEADER. The darker gray **background** color comes from HEADERSANDFOOTERS. Note in the second panel that inheritance works when ROWHEADER is specified with a *style-from* statement that points to the newly reformatted HEADER. REPLACE (from HEADERSANDFOOTERS), as expected, duplicates the output from the second panel in Table 11.

11) Style Element Inheritance from a Parent with a FROM Clause

Code

```
/*
The header background color comes from
HEADERSANDFOOTERS
*/
proc template;
  define style styles.from1;
  parent=styles.default;
/* from lineage #20 */
  style Header from HeadersAndFooters /
    font = (arial,12pt,bold)
    foreground=red;
  end;
run;
```

HTML Output

ROWHEADER Does Not Inherit from HEADER with STYLE HEADER FROM HEADERSANDFOOTERS

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Code

```
/*
In the second STYLE statement, the CHILD inhe-
rits from the newly formatted PARENT that was
a CHILD in the first STYLE statement.
*/
proc template;
   define style styles.from2;
   parent=styles.default;
 /* from lineage #20 */
   style Header from HeadersAndFooters /
     font = (arial,12pt,bold)
     foreground=red;
   style rowheader from header;
   end;
 run;
```

HTML Output

ROWHEADER Inherits from HEADER with STYLE ROWHEADER FROM HEADER

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Code

```
/*
HEADERSANDFOOTERS supplies the darker gray
background color to HEADER. With REPLACE,
ROWHEADER inherits the adjusted changes from
HEADER.
*/
proc template;
   define style styles.from3;
   parent=styles.default;
 /* from lineage #20 */
   replace Header from HeadersAndFooters /
     font = (arial,12pt,bold)
     foreground=red;
   end;
 run;
```

HTML Output

ROWHEADER Inherits from HEADER with REPLACE HEADER FROM HEADERSANDFOOTERS

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Child elements can also inherit directly from any lineage ancestor. In Table 12, HEADER and ROWHEADER background colors come from CONTAINER, the common ancestor.

12) Style Element Inheritance from an Early Ancestor with a FROM Clause

Code

```
/*
The HEADER and ROWHEADER background colors
come from CONTAINER, the common ancestor.
*/
proc template;
  define style styles.from4;
  parent=styles.default;
/* from lineage #20 */
  replace Header from Container /
    font = (arial,12pt,bold)
    foreground=red;
  end;
run;
```

HTML Output

HEADER and ROWHEADER Inherit From CONTAINER

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Occasionally *child* elements inherit from themselves. This type of inheritance occurs when a desired default setting is located in the child style element. Since no defaults are assigned to HEADER or ROWHEADER in Lineage #20, the DATA style element with two attributes shown in Figure 3 is being used instead.

Figure 3. Only the foreground color from the DATA style element is changed in Table 13 below. (As an aside, FONT from CONTAINER will also be changed).

Lineage#	Style#1	Style#2	Style#3	Style#4	Style#5	Style#6	Style#7
2	Container	Cell	Data	DataEmphasis	DataEmphasisFixed		

Detailed Listing for Lineage #2

Style	Default Assignment	Attribute
Container	**Abstract. Controls all container oriented elements.**	
	font = Fonts('DocFont')	FONT
	foreground = colors('docfg')	FOREGROUND
	background = colors('docbg');	BACKGROUND
Cell	**Abstract. Controls general cells.**	
Data	**Default style for data cells in columns.**	
	foreground = colors('datafg')	FOREGROUND
	background = colors('databg');	BACKGROUND
DataEmphasis	**Controls emphasized data cells in columns.**	
	foreground = colors('datafgemph')	FOREGROUND
	background = colors('databgemph')	BACKGROUND
	font = fonts('EmphasisFont');	FONT
DataEmphasisFixed	**Controls emphasized data cells in columns. Fixed font.**	
	font = fonts('FixedEmphasisFont');	FONT

In Table 13, the default DATA background color is being retained. It is the same shade of light gray that appears in all the data cells from the preceding table displays. However, the foreground (font) color is being changed from black to red, and courier-new is being used for the font-face.

13) Style Element Inheritance from SELF with a FROM Clause

Code

```
/*
The DATA background color comes from DATA
*/
proc template;
  define style styles.from5;
  parent=styles.default;
/* from lineage #2 */
  style data from data /
    font = ("courier new",10pt,bold)
    foreground=red;
  end;
run;
```

HTML Output

DATA Inherits From DATA

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

While counter-intuitive, it is possible for an ancestor to inherit directly from a descendent. In Table 14, ROWHEADER inherits from grandparent HEADERSANDFOOTERS but HEADER inherits attribute settings from its newly reformatted *child*, ROWHEADER. This is an unusual turn of events, because in conventional object-oriented languages such as C++, inheritance is exclusively one-way: from ancestor to descendent (or from base class to derived class).

14) Style Element Inheritance from a Descendent with a FROM Clause

Code

```
/*
For inheritance from child to parent to work
properly, the child element must be defined
BEFORE the parent element.
The default HEADER and ROWHEADER background
colors remain in effect.
HEADER inherits font settings from ROWHEADER.
*/
proc template;
  define style styles.from6;
  parent=styles.default;
/* from lineage #20 */
  style rowheader from headersAndFooters /
    font = (arial,12pt,bold)
    foreground=red;
  style header from rowheader /
    foreground=blue;
  end;
run;
```

HTML Output

ROWHEADER Inherits From HEADERSANDFOOTERS
Then HEADER Inherits from ROWHEADER

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

Also counterintuitive is the ability to program cross-lineage inheritance in ODS. In Table 15 HEADER and ROWHEADER in Lineage #20 inherit FONT, FOREGROUND, and the default setting for BACKGROUND from DATA in lineage #2.

15) Style Element Inheritance from a Different Lineage

Code

```
/*
Now, DATA, HEADER, and ROWHEADER have the same
format.
*/
proc template;
  define style styles.from7;
  parent=styles.default;
 /* from lineage #2 */
  style data from data /
    font = ("courier new",10pt,bold)
    foreground=red;
 /* from lineage #20 */
  style header from data;
  style rowheader from header;
end;
run;
```

HTML Output

Cross-Lineage Inheritance
HEADER and ROWHEADER Inherit From DATA

The FREQ Procedure

Product	Frequency	Percent	Cumulative Frequency	Cumulative Percent
Boot	864	30.44	864	30.44
Sandal	564	19.87	1428	50.32
Slipper	794	27.98	2222	78.29
Sport Shoe	616	21.71	2838	100.00

INHERITANCE: ODS vs C++

It is obvious from Tables 14 and 15 that there are significant differences between ODS and C++ when it comes to how classes (style elements) relate to each other. In C++ a new, derived class is created from an existing base class. The derived class incorporates all the features of the base class, and then adds needed embellishments [1, p. 333]. For example, CAR could be defined as a base class, and CHEVY with its particular logo and body design could be derived from CAR. Furthermore, CAR, with a few modifications, could be transformed into an ABSTRACT class. Then only CHEVY "objects" encompassing CAR properties could be created. CARs, as separate entities, could not come into existence on their own [1, p.504].

In the case of ODS, it is difficult to know if style elements are *classes* or *objects*, since associated attributes are assigned *values* in the default template. (Variables are defined for C++ classes. Objects come into existence when values are assigned to those variables). With blurred boundaries between *class* and *object* in ODS it is also difficult to understand just what is meant by ABSTRACT. Earlier it was shown that abstract style elements in ODS could be directly modified by substituting REPLACE for STYLE in a template definition. In C++ such a manipulation would not be possible, since objects cannot be created from abstract classes.

However, with the greater flexibility, it becomes possible to physically trace inheritance by lineage in ODS. In Table 16, the domains of the first five style elements in Lineage #20 are traced by adding **`borderwidth`** and **`bordercolor`** attributes to both abstract and regular style element definitions.

From Table 16, it can be seen that the tracings become more restricted as the lineage is traversed. CONTAINER, for example, encompasses both the BODY and CONTENTS windows (including slider bars) whereas ROWHEADER

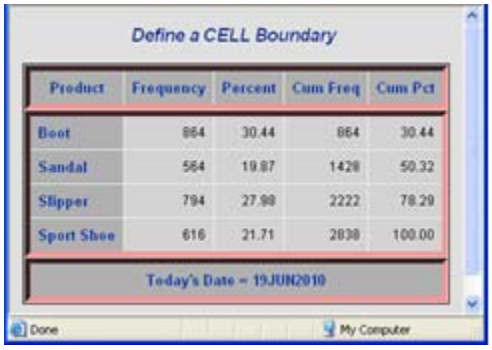
overlays just the first data column in a table that is viewed exclusively from the BODY window. Again, the contrast between ODS and C++ is noticeable. The base class in C++ is smaller, not larger, than its derived classes.

16) Output Mappings for the First Five Style Elements in Lineage #20

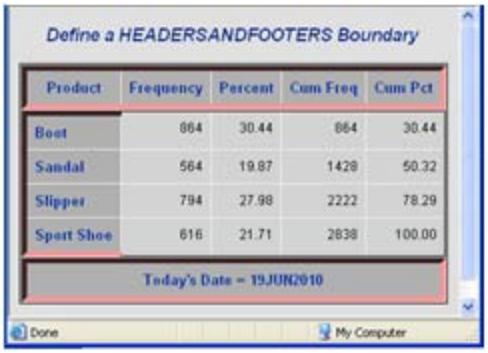
Lineage#	Style#1	Style#2	Style#3	Style#4	Style#5	Style#6	Style#7
20	Container	Cell	HeadersAndFooters	Header	RowHeader	RowHeaderEmphasis	RowHeaderEmphasisFixed

CONTAINER covers both CONTENTS and BODY windows including the slider bars.

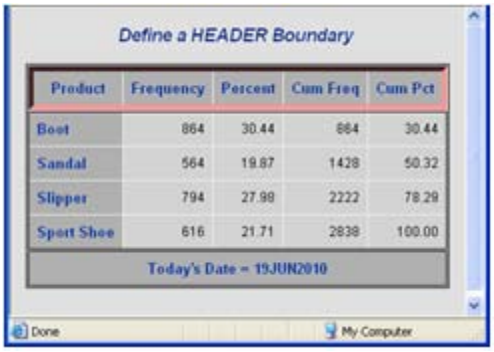
CELL covers HEADER, DATA and FOOTER



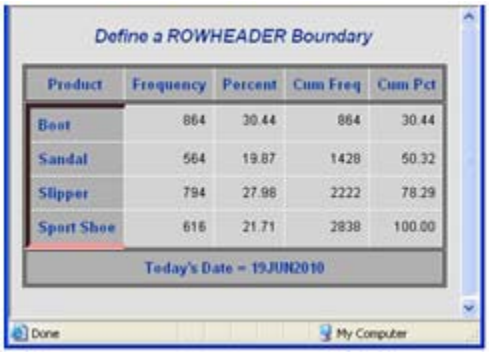
HEADERSANDFOOTERS covers HEADER, ROWHEADER and FOOTER. (Most of DATA is excluded).



HEADER covers HEADER only



ROWHEADER covers ROWHEADER only



SUMMARY AND CONCLUSIONS

Tracking inheritance in the ODS Styles.Default template becomes possible with assistance from a lineage tracer. With a lineage tracer abstract elements can be distinguished from their regular counterparts, ancestors can be readily identified, and in-lineage attributes can be separated from non-relatives. When such distinctions are made, guidelines can be established for working more effectively with style element modifications. For example, by checking the status of a particular style element in the lineage tracer, we learned that abstract elements could only be changed with REPLACE, whereas both STYLE and REPLACE work with different results on regular style elements.

With a lineage tracer it is also possible to verify that the FROM keyword is only needed when attribute defaults are to be transferred to an updated style element. Many examples that featured FROM highlighted the flexibility of ODS inheritance. Style elements can obtain default values from parents, distant ancestors, themselves and even their descendents!

It was also demonstrated in the paper that non-lineage attributes can be added to a style element in a new template. If successfully added, the new attribute can be passed on to descendents in accordance with the rules for inheritance. Unfortunately, however, outcome with non-lineage attributes can be unpredictable. Sometimes, as in the case of **`just`** and **`vjust,`** a conflict may arise between ODS and PROCEDURE settings. If such a conflict arises, the PROC always prevails. Non-lineage attributes illustrated in the paper include **`borderwidth, bordercolor, just, vjust, cellwidth`** and **`cellheight`** (See gray highlighted entries in Tables 9 and 10 for the last two).

Probably it is not a good idea to try to shoehorn ODS inheritance into the C++ object-oriented framework. ODS is a unique construct with its own rules for transferring information from one style element to another.

COPYRIGHT STATEMENT

REFERENCES

[1] Lafore, Robert. *The Waite Group® Object-Oriented Programming in C++ Second Edition.* Corte Madera, CA: Waite Group Press, 1995.

[2] Haworth, Lauren E., Cynthia L. Zender, and Michele M. Burlew. *Output Delivery System: The Basics and Beyond.* Cary, NC: SAS Institute Inc., 2009.

[3] SAS Institute Inc. *SAS® 9.1 Output Delivery System: User's Guide.* Cary NC: SAS Institute Inc., 2004.

[4] Watts, Perry. *Using Recursion to Trace Lineages in the SAS® ODS Styles.Default Template*. Proceedings of the 23rd Annual Northeast SAS Users Group Conference. Baltimore, MD, 2010, paper #BB13.

TRADEMARK CITATION

CONTACT INFORMATION

Please send comments and requests for the Lineage Tracer to mailto:perryWatts@comcast.net

APPENDIX

[1) ATTRIBUTE LISTING]

Attribute listing from the 9.1.3 ODS Manual (Gray Attributes are not found in the ODS STYLES.DEFAULT Template)

Attribute	Definition	Legitimate Values
ACTIVELINKCOLOR	Color for active links	Any valid SAS color
ASIS	How leading spaces and line breaks are handled	ON or OFF. If OFF then leading spaces are trimmed, line breaks ignored.
BACKGROUND	Background color	Any valid SAS color
BACKGROUNDIMAGE	Background image	'string' where string is the name of a GIF or JPEG file identified with a simple file name, a complete path, or a URL. Easiest approach: use a simple filename and place all image files in the local directory.
BODYSCROLLBAR	Scroll bar in the body file frame	YES | NO | AUTO, where AUTO specifies "only if needed".
BODYSIZE	Frame width for body file in HTML	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT) --OR-- integer% of entire display
BORDERCOLOR	Single border color	Any valid SAS color.
BORDERCOLORDARK	Darker color used in a 3-D 2-color border.	Any valid SAS color.
BORDERCOLORLIGHT	Lighter color used in a 3-D 2-color border.	Any valid SAS color.
BORDERWIDTH	Table border width	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
BOTTOMMARGIN	Bottom margin for a document	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
BULLET	Bullet string in the contents file	Bullets='string' where string = 'circle', 'decimal', 'disc', lower-alpha, lower-roman, 'none', 'square', upper-alpha, or upper-roman (I, II, III)
CELLHEIGHT	Cell Height	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT) --OR-- integer% of table height
CELLPADDING	Amount of white space surrounding text in a cell	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT) --OR-- integer% of table
CELLSPACING	Space between cells	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
CELLWIDTH	Cell width	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT) --OR-- integer% of table width
CONTENTPOSITION	HTML contents frame location	LEFT | RIGHT | TOP | BOTTOM Alias(L,R,T,B)
CONTENTSCROLLBAR	Scroll bar in the contents and page files	YES | NO |AUTO (AUTO means only when necessary)
CONTENTSIZE	Frame width for contents frame in HTML	nonnegative number (in pixels) --OR-- integer% of entire display
FILLRULEWIDTH	Overlay a rule onto the white space surrounding text in a cell	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
FLYOVER	Text shown in a tool tip for the cell	'string'
FONT	FONT= [face(s), size, keywords]. Keywords= font weight and font style. The destination device uses the first installed font	Syntax: "font-face-1 <... , font-face-n>", font-size, keyword-list myfont=("arial, helvetica", 4, bold roman) myfont=(Arial, 2pt, medium italic)
FONT_FACE	Font Face	font-face-1 <... , font-face-n>
FONT_SIZE	Font size	A nonnegative number + a unit of measurement OR relative size (range=1-7)
FONT_STYLE	Font style	ITALIC | ROMAN |SLANT
FONT_WEIGHT	Font weight	MEDIUM | BOLD most popular. Other choices exist.
FONT_WIDTH	Font width	Few fonts honor font widths (e.g. compressed, narrow).

Attribute	Definition	Legitimate Values
FOREGROUND	Foreground color	Any valid SAS color
FRAME	Frame definitions for an HTML table	ABOVE= BELOW= a border at top OR bottom. BOX =borders all around. HSIDES= VSIDES borders at top and bottom OR left and right side. LHS = RHS=a border at the left side OR right side. VOID= no borders.
FRAMEBORDER	Border for an HTML frame	ON | OFF
FRAMEBORDERWIDTH	Border width for an HTML frame	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
FRAMESPACING	Space between HTML frames	Integer only (no units of measurement defined)
HREFTARGET	Target window of link	_BLANK for new window. _PARENT for source window. _SEARCH for browser's search pane. _SELF for current window (default). _TOP for the topmost window. 'name' for a specified window or frame.
HTMLCLASS	Stylesheet class for table or cell	'string'
HTMLCONTENTTYPE	Content type for pages sent directly to a web server rather than to a file	'string'
HTMLDOCTYPE	Doctype declaration for an HTML document = opening "<!DOCTYPE" and the closing ">"	'string'
HTMLID	ID for table or cell	'string'
HTMLSTYLE	Individual attributes and values for table or cell	'string'
INDENT	Indention depth	n=number of spaces. Default: 2 for XML, 0 every-thing else
JUST	Justification	CENTER | DEC | LEFT | RIGHT (DEC means at decimal points)
LEFTMARGIN	Document left margin	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
LINKCOLOR	Unvisited link color	Any valid SAS color
LISTENTRYANCHOR	Turn link to a table of contents entry on or off.	ON|OFF
LISTENTRYDBLSPACE	Double space between entries in the table of contents	ON|OFF
NOBREAKSPACE	How spaces are handled.	ON: SAS won't break a line at a space character. OFF: SAS will break a line at a space character.
OUTPUTHEIGHT	Graphics Height in a document	n(in pixels)
OUTPUTWIDTH	Table Width	n(pixels) OR n% (of window)
OVERHANGFACTOR	Upper limit for column width extension	n (a factor). A value between 1 and 2 is typical.
PAGEBREAKHTML	HTML placed at page breaks	'string'
POSTHTML	HTML code that goes after table or cell	'string'
POSTIMAGE	Image after table or cell	'string' | fileref names a GIF or JPEG file. 'string' includes a simple filename, complete path, or URL.
POSTTEXT	Text after cell or table	'string'
PREHTML	HTML code before HTML table or cell	'string'
PREIMAGE	Image before table or cell	'string' | fileref names a GIF or JPEG file. 'string' includes a simple filename, complete path, or URL.
PRETEXT	Text before table or cell	'string'
PROTECTSPECIALCHARS	Interpret less-than signs (<), greater-than signs (>), and ampersands (&)	ON | OFF | AUTO ON means characters not part of HTML. OFF means they are.
RIGHTMARGIN	Right margin for a document	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
RULES	Lines within a table	ALL=between all rows and columns. COLS be-tween all columns. GROUPS between the table header and the table and between the table and the table footer, if there is one. NONE no rules anywhere. ROWS between all rows
TAGATTR	Text inserted in the HTML code	'string'

Attribute	Definition	Legitimate Values
TOPMARGIN	Top margin for a document	nonnegative number + a unit of measurement (e.g. IN, CM, MM, PT)
URL	Specify target URL	'Uniform-Resource-Locator'
VISITEDLINKCOLOR	Visited link color	Any valid SAS color
VJUST	Vertical justification	BOTTOM | MIDDLE | TOP
WATERMARK	Translate the target for BACKGROUNDIMAGE into a "watermark. A watermark appears in a fixed position as the window is scrolled	ON|OFF

2) COLOR TRACER (Sample Output. The complete listing is in the HTML file associated with the paper)

Color Tracer for the SAS STYLES.DEFAULT Container Lineages

COLORS: (FG=Foreground BG=Background)			Style Element
Location	Abbreviation	Color	(* = Inherited)
batchbg	bga3	cxd3d3d3	Batch
batchfg	fga1	cx000000	Batch
bylinebg	bga2	cxb0b0b0	Byline
bylinefg	fga2	cx0033aa	Byline
captionbg	bga	cxe0e0e0	AfterCaption
			BeforeCaption*
			Caption
captionfg	fga1	cx000000	AfterCaption
			BeforeCaption*
			Caption
conentryfg	fga2	cx0033aa	ContentItem
			FolderAction*
			IndexItem
			PagesItem*

3) FONT TRACER **(Sample Output. The complete listing is in the HTML file associated with the paper)**

Font Tracer for the SAS STYLES.DEFAULT Container Style Elements

Font	Font Name	Style Element (* = Inherited)
"sas monospace, courier new, courier, monospace",2	batchfixedfont	Batch
"arial, helvetica, sans-serif",3	docfont	Body
		BodyDate*
		ByContentFolder*
		BylineContainer*
		ColumnGroup*
		Container
		ContentFolder*

●●●

"arial, helvetica, sans-serif",4,bold	headingfont	AfterCaption
		BeforeCaption*
		Byline
		Caption*
		Footer*
		FooterEmpty*
		Header*
		HeaderEmpty*
		HeadersAndFooters
		RowFooter*
		RowFooterEmpty*
		RowHeader*
		RowHeaderEmpty*
	strongfont	DataStrong
		FooterStrong
		HeaderStrong
		PageNo
		RowFooterStrong
		RowHeaderStrong
"arial, helvetica, sans-serif",5,bold italic	titlefont	SystemFooter
		SystemTitle
"arial, helvetica, sans-serif",4,bold italic	titlefont2	ProcTitle
		TitlesAndFooters